Using a Fisheye View to Visualize Change-Over-Time in Support of Digital Forensic Examinations

Timothy R. Leschke
Timothy.Leschke@ManTech.com
Timothy.Leschke.ctr@dc3.mil
Tleschk1@umbc.edu

Penny Rheingans
Rheingan@umbc.edu

Alan T. Sherman
Sherman@umbc.edu

Abstract:

By understanding how digital evidence has changed-over-time, digital forensic examiners are better able to explain "what happened." We propose a data visualization technique that supports the examiner's understanding of how digital evidence has changed-over-time. By providing a means for the digital forensic examiner to perceive digital evidence in a visual format, the digital forensic examiner is able to conduct faster analysis, better comprehend data that has changed, and discover new relationships among the digital forensic data that were not previously known.

We develop what we call a "segmented box and whisker" visualization icon and apply it in a fisheye view, which allows the viewer to comprehend the content and context of a directory structure that has changed-over-time. We explore well known content+context visualization techniques and explain why a fisheye view is the best choice for displaying directory structure information. We develop a theoretical thresholding algorithm which reduces the dataset by hiding less important data. We provide a theoretical solution for dealing with scrolling issues and issues related to maintaining the focal point. We conclude with a theoretical data visualization technique which enhances one's ability to explain "what happened", which is the primary goal of every digital forensic examination.

1. Introduction:

Time is an illusion that is created by the mind's response to its perception of a world of continuous change. Time and change share a common quality that is often expressed as the single concept of *change-over-time*. For the digital forensic examiner, being able to understand change-over-time supports the goal of being able to explain "what happened." One way to enhance the forensic examiner's understanding of change-over-time is to allow him to perceive data visually.

We propose a method for presenting data to a digital forensic examiner such that he can easily perceive how a directory-tree structure has changed-over-time. Our novel contributions include 1) the development of a single visual representation which reflects changes to a directory-tree structure over-time, and 2) the application of the traditional content+context visualization technique, known as a *fisheye view*, to the new application domain of directory structure information.

1.1 Digital Forensic Data that Records Change-over-Time:

Digital artifacts and applications which maintain different versions of files as they existed at different points in time are said to record change-over-time. A log-based file system is one example of an application that provides this functionality. Log-based file systems, such as Sprite, Fossil, ILFS, LogFS, YAFFS, and others, maintain files in a circular log. When a file is changed, it is copied into memory before it is changed and written back to the log's next available (new) location. This process results in logs containing several versions of files. By comparing the new version of a file to one or more of the old versions, one can determine what has changed. The number of versions of a file is only limited by the size of the log and by how frequently the log is recycled.

Another application that maintains data that records change-over-time is the Volume Shadow Copy Service. Although this service was first introduced in Windows Server 2003, it is perhaps best known for being part of the Windows Vista and Windows 7 operating systems. This service creates backup copies of data so the user can restore individual files, or even restore the entire operating system, to the state that it was in at a previous moment in time. This is useful if the user wants to recover data that was accidentally deleted or modified.

The test data set that we used to support our visualization research was obtained from a Volume Shadow Copy Service. We installed Microsoft Office on a computer containing the Ultimate version of the Microsoft Vista operating system. We wrote a small C# program that traverses the entire directory tree and exports the meta-data for every directory along its path. This text output includes the name of the directory and information about each directory's parent-child relationship. We obtained directory-tree structure information from multiple time periods by accessing records that are maintained by the Volume Shadow Copy Service. We ensured there would be several time periods to work with by creating restore points 1) after installing Vista (but before activation), 2) after activating Vista, 3) after installing Service Pack 1, 4) after installing Microsoft Office, 5) after activating Microsoft Office, 6) after creating some user directories, and 7) after deleting some user directories. An eighth restore point was created automatically during the installation of the service pack. As all of the records were compiled into one file, the directory parsing tool added "true" and/or "false" values to each directory record to designate if the directory existed or not during each time period. PsExec was used to elevate the access rights for our directory parsing tool to *system level* which allowed us to obtain records for all *user*, *application*, and *system* directories. Our output file yielded approximately 700,000 directories!

1.2 The Motivation for Data Visualization:

Because more information can be obtained through vision than through all other senses combined, obtaining information through data visualization presents the greatest bandwidth for human perception. With data from personal computing environments and digital artifacts surpassing terabyte levels, the amount of data that is subject to a digital forensic examination has grown to almost unmanageable levels. One way to keep pace with the growing quantity of digital evidence is to increase the bandwidth by which digital forensic examiners perceive forensic data. The need for an *increased perceptual bandwidth* is one of the primary motivations for applying data visualization techniques to the domain of digital forensics.

Another motivation for applying data visualization techniques to digital forensics is *knowledge discovery* [1]. A query can be a very powerful technique, but a query can only be utilized if one knows exactly what one is looking for. Unfortunately, in the context of a cybercrime investigation, one often does not know what they are looking for. In this situation, a visualization technique is more useful because it displays data in a way that unique relationships among the data can be easily *seen* and discovered. For example, the digital forensic examiner may benefit by *seeing* which time-period contains the greatest amount of change because this time-period might contain the greatest amount of usable evidence.

A third motivation for utilizing data visualization is its ability to *increase productivity* by speeding-up analysis. Testing has shown that software engineers that make use of data visualization techniques have an increased productivity [1]. We believe an increase in productivity will also be actualized by digital forensic examiners if they utilize more data visualization techniques.

A fourth motivation for data visualization is its support of better comprehension by the viewer. For example, consider the use of the Unix *diff* command for comparing two files [1]. Although all of the differences between two files are expressed in the *diff* command's textual output, this textual output is not easily comprehended by the user. It is easier for the user to comprehend the differences between the two files if the *diff* command's output is displayed visually [1].

2. Development of the "Segmented Box and Whisker":

We have developed a visualization icon called a *segmented box and whisker* which we use to represent a file-system directory. The *segmented box* portion of the icon is a rectangle that is divided evenly into segments which represent time-periods that are ordered from left to right; from the most-distant time-period to the most-recent time-period. Each segment is colored either yellow or red depending on if the directory exists or does not exist during that particular time period.

The *whisker* portion of the *segmented box and whisker* icon is a thin black line that extends to the left of the *segmented box.* This whisker is used to denote the parent-child relationship by indenting the child directory under the parent directory. Figure 1 provides an example of three *segmented box and whisker* visualization icons that are used to represent file-system directories that have parent-child relationships. Figure 1 shows the CountryMusic directory is indented below the MyMusic directory, and the MyMusic directory is indented below the MyDocuments directory. Using whiskers to express indentations allows each *segmented box* to be arranged such that the segments from the same time-period are ordered in columns. This allows the viewer to easily scan a column and logically associate all of the directories that exist during a chosen time-period. For example, the far right column in Figure 1 shows that the directories named MyDocuments and MyMusic exist during that time period while the CountryMusic directory does not.

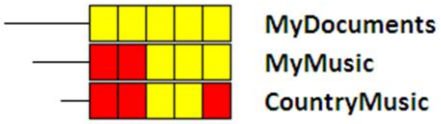

Figure 1

The use of yellow and red to denote the existence and non-existence of a directory is a technique known as *layering information* [1]. This technique makes it easy for the viewer to logically focus on one subset of data over another by concentrating on one color over the other. An additional benefit to layering information is that it makes it very easy for the viewer to identify where yellow and red segments border each other. Easily identifying these borders is important because it helps the viewer identify where change has taken place. Our choice of red to denote the non-existence (or death) of a directory is culture-based. Other cultures, like China's, represent good fortune with red, while green denotes death [13].

The development of the *segmented box and whisker* marks the completion our first goal, which is to create a visualization that portrays how a single directory changes over time. Our second goal is to present this information within the global context of all of the directories within the entire directory tree. Our second goal is achieved by implementing a *segmented box and whisker* within a *fisheye view.*

3. Visualizing Context with a Fisheye View:

Our objective is to create a single picture that conveys how a directory has changed over time, while also conveying an understanding of the global context of the directory. Our challenge is in displaying an enormous data set that contains about 700,000 data points. The amount of computer screen real estate is the limiting factor. If one assumes that we have access to a high-end monitor (like one that supports medical imaging, with 4096 x 2160 pixels), we are limited to displaying at most 4,096 items in a column, where each item has the height of just one pixel. We can fit 700,000 items within the boundaries of this high-end monitor if we arrange the items in a checker-board format and each item is comprised of about twelve pixels. However, we believe twelve pixels is too small to support a visualization that is useful for the user. This approach also scatters the icons all over the screen, which challenges the viewer's ability to grasp the global context. Therefore, displaying 700,000 icons on the computer screen at the same time is not a viable option.

S*crolling* [8] is also not acceptable for our needs because it forces the viewer to commit all previously viewed screen images to memory so that the entire data set can be mentally reconstructed from its parts, and the specific item can be comprehended within the context of the entire data set. We believe this approach exceeds the cognitive capacity of the viewer.

Magnification is another approach to viewing items within a global context that is not suitable for our needs. The two types of magnification to consider are *linear* and *non-linear.* Linear magnification is the type that one experiences when one looks through an ordinary magnifying glass [5]. The challenge with this approach is that there are *two cognitive/perceptual levels* which the viewer must mentally tie

together [5]. Within the magnified area, the data is presented in greater perceptual detail, whereas the non-magnified area is presented in less perceptual detail. The difference between these two levels of perceptual detail makes it difficult for one to mentally map one level of detail to the other. Furthermore, linear magnification creates an area that appears to *float* above the non-magnified area [4]. In addition, the boundary between these two levels of magnification is usually distorted [8]. Finally, linear magnification usually creates an *occlusion*, which is the blocking of neighboring areas of the non-magnified area with the magnified area [5]. Thus, linear magnification leads to some of the data being lost, which leads to a loss of the global context [11].

In a non-linear magnification, the change from one magnification level to the next is not an abrupt change as with a linear magnification. The change is more gradual, but it still leads to a distortion. The distortion is caused by the visualization being stretched between the lower magnification level and the higher magnification level. There is no true *occlusion* as with linear magnification. However, the distortion that is caused by the stretching effect of non-linear magnification behaves just like an occlusion in that the data is not easily perceived by the viewer.

Another visualization technique that is similar to magnification is *panning* and *zooming*, or as some call it, *pan-zoom* [3]. Panning over and entire data set allows one access to the global context of the data, whereas a zoom into a specific subset of the data allows more detail to be understood. A limitation to the *pan-zoom* approach is that panning requires a linear transformation whereas zooming is a logarithmic transformation [3]. Requiring the viewer to comprehend these two scales at the same time does not support easy perception. Furthermore, *pan-zoom* can lead to the loss of the overall structure or global context [12].

Unlike zooming, the *semantic zooming* approach allows a data point to change its appearance as the amount of available screen real estate changes [3]. For example, a page of text could be represented as a *point*, then a *solid rectangle*, and finally a *full text page* as the amount of room to display it increases [3]. A similar approach has been implemented for the representation of castles [4]. *Semantic zooming* is unacceptable for our needs because it allows a data point to change its appearance, which we believe distorts the information being perceived.

In our attempt to represent our data set of 700,000 items to the viewer, we have considered *scrolling*, *linear* and *non-linear magnification*, *pan-zoom*, and *semantic zooming*. Each of these visualization approaches is inadequate for their own reasons. We consider classical focus+context visualization approaches to help us meet our goal.

3.1 Classical Focus+Context Visualization Approaches:

In a classical focus+context display, one is allowed to focus on one part of a data set while retaining the global context provided by the remainder of the data set. This is precisely what we aim to do when we visualize the change of a particular directory within the context of the entire directory tree structure. We investigate many classical focus+context display techniques before concluding that the *fisheye view* is the best match for our visualization needs.

Polyfocal Display:

Polyfocal display, or *polyfocal projection* [4], balances the effect of applying a magnification function to one part of a display by applying a *negative magnification* [8] function to another part of the display. This produces areas that appear to vanish from the display. We reject this approach for visualizing our data set because by causing data to vanish for the display, we believe the global context is lost, or at least, is severely distorted.

Perspective Wall:

The *perspective wall* [9] integrates detail and context as it projects data onto the panels of a three-faced wall. The middle panel presents a detailed representation of the data of interest. The side panels, which angel away from the viewer at about forty-five degrees and also appear to extend to infinity, project the remainder of the data in less detail.

We reject the *perspective wall* because the side panels that extend to infinity distort the global context by visually reducing the distant data points to nothing. We maintain that distorting the global context of digital forensic data can lead to incorrect analysis, and eventually, incorrect conclusions. We also reject the *perspective wall* because it only works with *linear* data sets [12]. It does not work with our directory structure data set that extends in both directions like a rectangular array [11].

Bifocal Display:

Bifocal display arranges the data representations into three vertical strips [12]. A detailed representation gets displayed when the data point is scrolled into the middle strip, and a distorted representation gets displayed when the data point is scrolled into one of the two outside strips. Bifocal display has been called a special case of the *perspective wall* [8]. We reject the *bifocal display*, because like the *perspective wall*, it too distorts data which can lead to incorrect analysis and incorrect conclusions.

The Document Lens:

Document lens displays the data of interest in a rectangular focal point in the middle of the screen, similar to the *perspective wall*. But unlike the *perspective wall* which wastes space by not utilizing the area above or below the middle panel [8], the *document lens* utilizes this screen real estate by allowing the user to scroll up and down as well as right and left. However, the ability to move the rectangular focal point in four directions is only useful for domains that are rectangular, like text documents. When viewing text, a technique known as *greeking text* can be employed to abstract the data that lies outside the focal point [11]. Greeking text involves the replacing of text with a simple line.

We reject the *document lens* for our problem domain because the data that is outside of the rectangular focal point is distorted, which we want to avoid. We also reject the *document lens* because we contend the viewer cannot comprehend the global context of a data set that appears to expand in four directions.

Cone Trees:

A *cone tree* is a three-dimensional visualization in which the parent node is represented as the apex of a cone and the children are positioned in a circle in the plane that makes up the rim of the cone [9][12]. *Cone tree* representations are truly three-dimensional in that nodes that are further away from the center of focus, or what some call the *synthetic camera* [12], are smaller. Nodes that are closer are portrayed as larger. *Cone trees* can be transparent, which helps the viewer see through the objects in the front in order to see an object in the back of the visualization. However, even with this use of transparency, *cone trees* are still limited to visualizing data sets with a maximum size of about a couple hundred items [12]. A *cone tree* is also only ideal for viewing data sets that can naturally be represented as node trees where there are parent nodes and child nodes. Although we might be able to represent a directory structure as a *cone tree*, we reject the *cone tree* approach because of its inability to work with a large enough data set.

Hyperbolic Browser:

A *hyperbolic browser* lays out data along a *hyperbolic plane* [7], which is a curved three-dimensional surface, such as the sides of a cone or a section of the surface of a sphere. Because the area of the hyperbolic plane naturally expands as one moves away from the center, projecting data sets onto this plain is ideal for those data sets that naturally expand, like out directory-tree structure that tends to expand into more child directories as one traverses further away from the root directory. Unfortunately, our data set is not a traditional directory-tree structure in that it includes information about how the directory-tree has changed-over-time. Our need to logically associate directories from the same time-period makes the scattering of data effect caused by the *hyperbolic browser* to be unsuitable for our needs.

Treemaps:

Treemaps are extremely powerful because they make use of all of the available screen real estate. Their natural application domain is hierarchical data that is arranged in trees [1], much like our directory-tree structure data. The *treemap* itself is a visualization that divides the available screen real estate into sections whose surface areas are proportionate to the amount of data that each data point represents. This space-saving approach leads to an efficiency that has allowed over a million data points to be represented by a single *treemap*. Although the nodes within hierarchical data are easily displayed, the actual structure of the hierarchical data is obscured. Thus, *treemaps* are not suitable for our problem domain which requires the expression of hierarchical data. The visual scattering effect caused by a traditional *treemap* implementation also makes them inappropriate for trying to logically associate all directories from a common time period.

3.2 Fisheye View:

In a *fisheye view*, nearby items are presented with great detail whereas items that are further away are presented with less detail [2]. This approach supports visualizing local detail within a global context. This approach is also a natural approach for displaying hierarchical file systems [2]. Although some have cited instances in which a *fisheye view* has lead to a distortion [5][12], in the original design, distortion is avoided by allowing data to be either completely present or completely absent [12]. Because a *fisheye view* 1) does not distort data, 2) works well with hierarchical data, 3) expresses both content and context information, and 4) can be applied to large data sets, it is an appropriate approach to apply to our problem domain.

3.3 Applying a Fisheye View to a set of Segmented Box and Whiskers:

There are two ways to apply a *fisheye view* to a set of *segmented box and whisker* icons. One way is to reduce the height of the *segmented box and whisker* so that it requires less space along the Y-coordinate plane. This is similar to *non-linear magnification* [6]. By only reducing the height of a *segmented box and whisker*, segments from the same time period remain aligned vertically which allows for their logical association by the viewer. The other way to apply a *fisheye view* is to remove the entire *segmented box and whisker* completely. This approach is in agreement with the original *fisheye view* in which data can be completely absent [12]. We propose using both approaches.

The method by which we choose to represent a data point is provided by a *thresholding* algorithm [8]. *Thresholding* involves the assignment of a *threshold value* to data points that reflect their relevance, or as some call it, their *degree of interest* [2] or *visual worth* [12]. This *threshold value* is then used to determine if a data point is represented in great detail, less detail, or is hidden through *selective omission* [9]. Our novel contribution to this problem domain is the recognition of the natural *thresholding value* that is implied within a directory-tree structure. We have identified four (4) rules for assigning *threshold value* relevance; 1) directories that are located within a directories file-system path are the most relevant, 2) directories that add to an understanding of the *global* context are the second most relevant, 3) directories that add to an understanding of the *local* context are the third most relevant, and 4) all other directories are less relevant.

To illustrate our *thresholding* algorithm, we present the following hypothetical situation (Figure 2). Suppose someone has created a directory (named *MySecretDirectory*) for hiding files, and this directory is located at /Program Files/Microsoft Office/Office12/MySecretDirectory. According to our first rule, this directory information is the most relevant and it must be included in the visualization. Figure 2 shows the “Program Files,” “Microsoft Office,” “Office12”, and “MySecretDirectory” directories are all displayed in fulfillment of rule 1. Our second rule requires us to display directories that add an understanding of the *global* context. We contend that the directories that provide the most global context are those that are closest to the root, while also offering a general understanding of the entire directory tree. The directories within Figure 2 that accomplish this are the “$Recycle.Bin,” “Boot,” “Documents and Settings,” “MSOCache,”, “PerfLogs,” “ProgramData,” “System Volume Information,” “Users,” and “Windows” directories. Our third rule requires us to add directories that contribute to the understanding of the *local* context. We have accomplished this in Figure 2 by adding the directories that

are the siblings of “MySecretDirectory.” These directories are expressed as *segmented box and whisker* icons that have been reduced in height and their names have been hidden. This allows the sibling directories to add local context without obscuring an understanding of the directory at the focal point, “MySecretDirectory”. The remaining directories from out original set of 700,000 directories are hidden completely because they do not add to the *global* or *local* context.

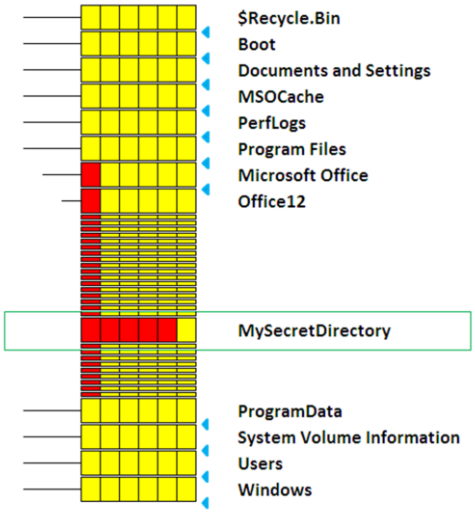

Figure 2

We recognize that the needs of the user cannot be known ahead of time. Therefore, it is not possible to hard-code the *thresholding* algorithm nor assign specific *thresholding* values to individual directories. We propose the use of *sliders* to allow the user to increase or decrease *thresholding* values so that the viewer can adjust the visualization to best suits their needs. A user will be able to increase the amount of *global* context that is expressed in order to support a general search, or increase the amount of *local* context to support a focused examination.

Figure 2 shows blue triangles which we call *expansion markers*. These triangles point below a *segmented box and whisker* icon if the corresponding directory can be expanded to expose child directories. These *expansion markers* are necessary to help the viewer understand where there is hidden data. In areas where a triangle *expansion marker* does not fit, we propose the use of a blue hash mark (—).

Figure 2 also shows a green rectangle called a *focal box*, which helps the viewer keep track of the focal point. This approach is more desirable than placing a reference mark in the margin [6], which can draw the viewer’s attention away from the data being viewed. Adjusting the *focal box* by *scrolling* through the data can cause the viewer discomfort or disorientation [8][1] because the visualization can appear to flash, shake, or jump-around slightly, much like how a video camera can produce a distorted picture when panned at high speed [8]. We contend that we can minimize the distortion caused by *scrolling* by minimizing the amount of re-drawing, which we control trough the *thresholding algorithm*.

4. Future Work:

We have developed an application for extracting information regarding the changes to a directory-tree structure. We have used this information to print *segmented box and whisker* icons for all of the directories in our data set. Because of the enormous size of this data set, our current visualization scrolls off the computer screen, which is the primary problem that needs to be addressed.

Our future work includes implementing the *thresholding algorithm*. We will implement the algorithm to allow the user to use *sliders* to adjust the *relevance values* of the directories being displayed. We will implement the *expansion markers* and the *focal box*. We will fine-tune the *thresholding algorithm* so as to minimize the distortion that might be caused by *scrolling*.

Once our visualization approach is implemented, we will conduct human-computer interaction experiments to determine the strength and weaknesses of the visualization when used by real human subjects. We will attempt to measure the ease at which people can use the tool. We will evaluate different color choices and sizes for the *segmented box and whisker*. We will evaluate if our visualization leads to 1) faster analysis, 2) better comprehension, and 3) the discovery of new relationships.

After we have developed a technique for visualizing change-over-time for a directory-tree structure, we will investigate visualizing change-over-time for directory *content* and change-over-time for *file attributes*. Our final goal will be to develop a way to visualize change to 1) *directory structure*, 2) *directory content*, and 3) *file attributes*, all in one visualization. We expect our challenge will be to blend the three visualization techniques into one such that they do not compete with each other.

5. Conclusion:

We proposed a data visualization technique that supports the examiner's understanding of how digital evidence has changed-over-time. By understanding how digital evidence has changed-over-time, digital forensic examiners are better able to explain "what happened."

We developed what we call a *segmented box and whisker* visualization icon and explain how it can be applied in a *fisheye view*, which allows the viewer to comprehend the content and context of a directory structure that has changed-over-time. We explored well known content+context visualization techniques and explained why a *fisheye view* is the best choice for displaying directory-structure information. We developed a theoretical *thresholding algorithm* which reduces the problem set by hiding less important data. We provided a theoretical solution for dealing with scrolling issues and issues related to maintaining the focal point. We conclude that we have developed a theoretical data visualization technique which enhances the examiner's ability to explain "what happened", which is the primary goal of every digital forensic examination.

References:

[1] Ball, Thomas, and Eick, Stephen. Software Visualization in the Large. Computer, Apr 1996, Volume: 29 Issue: 4, pages 33 – 43.

[2] Furnas, George W. Generalized Fisheye Views. Human Factors in Computing Systems CHI '86 Conference Proceedings, 16-23, 1986.

[3] Furnas, G.W. and Bederson, B.B. Space-Scale Diagrams: Understanding Multiscale Interfaces. In Proceedings of CHI. 1995, 234-241.

[4] Keahey, T. Alan. The Generalized Detail-in-Context Problem. Proceedings of the 1998 IEEE Symposium on Information Visualization (1998).

[5] Keahey, T. Alan and Robertson, Edward L. Techniques for Non-Linear Magnification Transformations. In Proceedings of the IEEE Symposium on Information Visualization, IEEE Visualization, pages 38-45, October 1996.

[6] Keahey, T. Alan, and Marley, Julianne. Viewing Text with Non-linear Magnification: An experimental Study. Technical Report 459, Department of Computer Science, Indiana University, April 1996.

[7] Lamping, John, Rao, Ramana, and Pirolli, Peter. A Focus+Context Technique Based on Hyperbolic Geometry for Visualizing Large Hierarchies. In Proc. ACM Conf. Human Factors in Computing Systems, CHI (1995), pp. 401-408.

[8] Leung, Y.K. and Aerley, M. D. Aerley. A Review and Taxonomy of Distortion-oriented Presentation Techniques. ACM Transactions on Computer-Human Interaction, Vol. 1, No. 2. (1994), pp. 126-160.

[9] Mackinlay, J. D., Robertson, G. G., and Card, S. K. The Perspective Wall: Detail and Context Smoothly Integrated. Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI '91); 1991 April 27 - May 2, 173-179.

[10] Munzner, Tamara, and Burchard, Paul. Visualizing the Structure of the World Wide Web in 3D Hyperbolic Space. Proceedings of VRML '95, (San Diego, California, December 14-15, 1995), special issue of Computer Graphics, ACM SIGGRAPH, New York, 1995, pp. 33-38.

[11] Robertson, Georeg G. and Mackinlay, Jock D. The Document Lens. In UIST '93: Proceedings of the 6th annual ACM symposium on User interface software and technology (1993), pp. 101-108.

[12] Sarkar, Manojit and Brown, Marc. Graphical Fisheye Views. Communications of the ACM. Volume 37 Issue 12, Dec. 1994.

[13] Ware, Colin. *Information Visualization, Perception for Design (Second Edition).* Elsevier, San Francisco, CA (2004). ISBN 978-1-55860-819-1.